INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. : (3) 954 90 20

Rapports de Recherche

N° 433

COMPUTATIONAL PROBABILITY AND ANALYSIS OF ALGORITHMS

Micha HOFRI

Juillet 1985

Computational Probability and Analysis of Algorithms

Micha Hofri
Department of Computer Science, The Technion
Haifa Israel

ABSTRACT

Three levels of interaction between numerical computing and probability will be presented:

a) Monte-Carlo computations, being the exploitation of a probabilistic approach to estimate an essentially deterministic result.

b) Numerical computation as an aid to conventional stochastic analysis (via queueing theoretical examples).

c) Computational procedures as generators of stochastic processes that have considerable interest. Their treatment - conventionally called analysis of algorithms - is a source of numerous new problems. Personal preferences dictate that most of the walk will be devoted to one or two (time permitting) examples of this nature. Specifically, a distributed algorithm used to resolve conflicts in a local computer communication network is investigated to find the maximum level of traffic it can tolerate; then we consider one-dimensional bin packing algorithms and present some asymptotic results.

RESUME

Trois niveaux d'interaction entre calcul numérique et calcul des probabilités sont présentés dans ce rapport :

a) Des calculs de types Monté-Carlo, qui sont le résultat d'une approche probabiliste pour l'estimation d'un phénomène essentiellement déterministe.

b) Des calculs numériques considérés comme un support à l'analyse stochastique classique, via des exemples théoriques de files d'attente.

c) Des procédures numériques considérées comme générateurs de processus stochastiques très importants. Leur traitement - usuellement dénommé analyse d'algorithmes - est la source de nombreux problèmes neufs. L'auteur, par goût personnel, consacre l'essentiel de ce rapport à deux exemples de ce genre. D'abord un algorithme distribué - utilisé pour la résolution des conflits dans un réseau local - pour lesquels on obtient le trafic maximum admissible ; ensuite un algorithme de "bin-packing" à une dimension pour lequel on présente quelques résultats asymptotiques.

COMPUTATIONAL PROBABILITY AND ANALYSIS OF ALGORITHMS

Micha HOFRI
Department of Computer Science, The Technion
Haifa Israel

ABSTRACT

Three levels of interaction between numerical computing and probability will be presented :

a) Monte-Carlo computations, being the exploitation of a probabilistic approach to estimate an essentially deterministic result.

b) Numerical computation as an aid to conventional stochastic analysis (via queueing theoretical examples).

c) Computational procedures as generators of stochastic processes that have considerable interest. Their treatment -conventionally called analysis of algorithms- is a source of numerous new problems. Personnal preferences dictate that most of the talk will be devoted to one or two (time permitting) examples of this nature. Specifically, a distributed algorithm used to resolve conflicts in a local computer communication network is investigated to find the maximum level of traffic it can tolerate; then we consider one-dimensional bin packing algorithms and present some asymptotic results.

1. INTRODUCTION

Computers and scientific work in probability -essentially applied probability- are related in a number of ways. My purpose here is to illustrate very briefly a few of these and to dwell in some more length on two examples that typify the relation which is to me the most enjoyable.

As we shall see, more than the trivial statements that computers produce numbers and probability is the branch of mathematics dealing with numerical measures play a role here. The first area to be presented exploits the capability of the computer to perform repetetively the same task a large number of times; the second indeed uses the computer as a number cruncher whereas when we come to the third area -the investigation of the stochastic properties of computations we turn the tables on the machine, though as we show these researches are highly symbiotic : as the machine is investigated it is also used in the research, and in more than one way.

2. SIMULATION

Simulation is a term currently employed to cover a vast variety of computing techniques. One gets an impression of this variety by browsing through simulation conference proceedings our at a few numbers of the journal that carries this name. Most commonly simulation denotes a computation that displays some features which make it possible for us to say that it "simulates" the evolution or characteristics of some "target system". From our point of view the interesting simulations are those that are driven by sequences of "random variables", in effect, sequences of numbers produced by pseudo-random number generators. The design and analysis of these generators is a fascinating field of study in its own right combining aspects of statistics, number theory and computer arithmetics. See Fishman (1978), Knuth (1972).

It appears entirely natural to use such simulation to observe estimates of statistics generated by processes for which our analytic tools are not sharp enough (e.g. a queueing system with resource constraints). It is equally useful however in situations where there is no underlying stochastic process. The classic example is the evaluation of the integral

$$(1) \qquad I = \int_a^b f(x)dx$$

where $f(x)$ is known to satisfy

$$A = 0 \leq f(x) \leq B \quad a \leq x \leq b.$$

Note that (1) can be rewritten as

$$(2) \qquad I = \int_a^b \int_A^B \chi\{y \leq f(x)\}dydx$$

with $\chi\{E\}$ the indicator function of E.

This suggest a natural way to evaluate I, using random variables : generate a sequence of pseudo-random numbers $\{u_j\}$, $j \geq 0$, treat them as i.i.d. U(0,1) and create the sequence $\{(x_j,y_j)\}$ with

$$x_j = a + \frac{u_{2j}}{b-a}, \quad y_j = \frac{u_{2j+1}}{B}$$

Having thus generated say n pairs, all we need do is count the number of pairs that are in the set $\{(x_j,y_j \mid f(x_j) \geq y_j,\ 1 \leq j < n\}$. If this number is m, than $I = B(a-b)m/n$ is our estimate of I. Obviously various refinements and generalizations are not hard to produce but we wanted just to point out a relation between stochastic computation and deterministic objectives. Recently probabilistic algorithms have been invented for determination of the primality of large integers, connectivity of graphs, maximal values of functions and similar "deterministic" properties.

3. THE COMPUTER AND QUEUEING THEORY

What does "solving a queueing problem" mean ? Like other culture-bound phenomena it depends both on the type of the problem and the time when the solution was obtained.

For certain problems - the simplest ones - we imply obtaining a complete specification of the distribution of the variables of interest, such as queue length and sojourn times, as a function of the time since a given state prevailed. More common is the situation where we find an underlying Markovian structure and proceed analytically to obtain explicit expressions for functionals of the <u>limiting</u> (or stationary) distribution of that Markov chain or process, and the problem is deemed solved.

The phrase "explicit expressions" is relative as well. For the M/M/1 queueing system we can manually evaluate almost any desirable functional of the limiting distributions. This is nearly true also for the GI/M/1 queueing system, but only up to a number x, defined as the solution of the (usually) transcendental equation

$$x = L(\mu - \mu x) \tag{3.1}$$

where μ is the service rate, L is the Laplace-Stieltjes transform of the interarrival time distribution and the solution must satisfy $0 < x < 1$.

Typically we use some standard numerical computer package to determine x. For more complex problems we are glad if we manage to come out with transforms of the distributions of interest. Inverting these transforms is often a formidable problem, usually only numerical approaches are possible, and even this often turns out to be a delicate affair. Several techniques exist, which usually call for extensive computations. Thus the analyses made during the '30s by Pollaczek and his school of the simplest single server system but under no distributional assumptions remained for a long time hidden behing the so called "complex transform smoke screen", at least as for as engineering usage is concerned.

The following scheme is familiar to anyone concerned with queueing theory:

(1) Find a (possibly partial) Markovian state descriptor
(2) Derive Chapman-Kolmogorov equations for the evolution of the chain or process defined in (1).
(3) Under the assumption that the system is stable, obtain the equations the stationary distribution of the process must satisfy.
(4) Rewrite the stationarity equations in terms of transforms and generating functions. When this is infeasible it is often also the end of the road for this problem, except when the state space is finite (and rather small).
(5) Solve the equations for the desired function(s).
(6) The solution obtained will nearly always depend on some unknown constants and simpler functions, that need be determined by indirect arguments (i.e. : not explicitly from the equations). These arguments often require solution of polynomial or transcendental equations, matrix manipulation etc. in a word, access to a computer.
(7) Moment evaluation by differentiating the transforms at suitable points.
(8) The expressions obtained for the moments and probabilities of interest are commonly involved and opaque. To obtain a more direct comprehension of the performance of the systems, and its dependence on the various parameters we return to the computer to manufacture tables and graphs.

As times goes on researchers tackle ever more ambitious and intricate problems, computers and decent numerical techniques become increasingly available with the obvious results. An interesting facet of this accompanying development is the approach - advocated and to a considerable extent pioneered - by Marcel Neuts, who holds that explicit expressions are not essential, that there exist problems that are best approached "algorithmically". This implies that the analyst does not- or cannot proceed to obtain the quantities he wants as an explicit expression. Often it is not even possible to find an implicit one - i.e. an equation - satisfied by the functions he is after, but the researcher can deduce an algorithm, specified usually in terms amenable to computer execution, whether numerical or symbolic , that will yield the desired results. (See Fayolle at al. 1985).

Let me finally mention another service rendered to the analyst by numerical computations, and that is the weeding out of errors in analysis. An example would serve well here : the performance analysis of the main memory subsystem in a certain computer organization boiled down to estimating queue lengths and waiting times in a one-server system, with constant service time, subject to batch arrivals with a rather complicated dependence structure. Since both FCFS and LCFS are viable service policies there, they were both analyzed. The two analyses required quite different methods and although the queue length process under both regimes is the same, there was no hope of directly comparing the expressions obtained even for the mean queue length. After the work was essentially completed I proceeded to compare them numerically, and in the process of a few weeks serveral very surprising errors were unearthed, some of which were quite subtle; it is difficult to imagine a more effective tool for this purpose.

4. ANALYSIS OF ALGORITHMS

One of the most impressive, and personally most satisfying phenomena accompanying the growing prominence of computers in our life, has been the emergence of new domains in mathematics and hosts of new concerns and problems in existing ones, all largely motivated by the nature of automatic computation.

The activity we now address is the direct investigation of these computational processes, or rather their quantification. The common appellation for this investigation is Analysis of Algorithms. An algorithm is a computational procedure for the performance of a well defined task such as

- sorting a sequence of numbers,
- solving a linear programming problem,
- printing a line,
- transfering the contents of a file between two computers,
- retrieving structured information from a data base, etc.

An algorithm specifies :

- the operations that need take place,
- termination conditions.

The notion of variability enters through the requirement that the algorithm shall process correctly all input data that satisfy the initial constraints. Typically different initial data will require different sequence of operations, produce different results and consume different amounts of resources such as time and storage.

The activity of the design of algorithms is largely involved in finding optimal algorithms. The analyst, on the other hand considers a given algorithm, which is often not optimal but interesting or satisfactory on other grounds.

The analysis can be done, grosso modo, in two directions :
Worst case analysis : considering all valid input data, how bad can the algorithm be, in terms of the end result or the resources consumed in the process.
Stochastic analysis : assigning a suitable measure to the space of valid data, how are the various characteristics of the algorithm distributed.

The first type provides a performance guarantee. The second is often of interest when repeated use of the algorithm is envisioned and representative estimates of what we may expect of its behavior is of more relevance.

The first algorithm that we show provides a good example of this dichotomy.

4.1 - Bin Packing

The problem of bin packing has the following form :

Given a list of "pieces" of lengths $\{\chi_i, 1 \leq i \leq n\}$, where $\chi_i \in [0,1]$ and an adequate supply of "bins" of size 1, pack the pieces into the bins so that none overflows.

The classical version asks to use the least number of bins possible, and apparently no algorithm can do this essentially faster than dynamic programming, which is extremely slow here.

Thus cheaper, though suboptimal algorithms are of interest. A simple candidate is Next Fit, which packs the pieces in the order they occur in the list like this :

1) let $j = 1$,
2) Find the largest k such that $\sum_{i=j}^{k} X_i \leq 1$
3) Pack pieces j to k into a bin, discard it.
4) Set $j = k+1$ and return to 2), unless the list is exhausted.

NF does nothing clever ; still, we want to analyze its performance. Let $L = \sum_{i=1}^{n} X_i$. It is not hard to find examples where the ratio NF_n/OPT_n is arbitrarily close to 2, where A_n is the number of bins used by algorithm A. This is its worst case behaviour. What can wa say about its expected behviour ?

Stochastic Analysis of NF :

Picking the easiest case we assume $X_i \sim U(0,1)$, independently. It has been shown that $E(OPT_n) \to E(L)$, that is, in the limit the optimum packing is dense, so we are interested in the ratio $E(NF_n)/E(L)$, with $E(L) = n/2$ here.

One approach is to consider the level reached by the packing at step 3 of the algorithm. These levels at successive bins form a Markov chain, and it is not difficult to show that the chain approaches a stationary distribution with mean $b = 0.75$, hence

$$E(NF_n)/E(L) \to 1/b = 4/3, \text{ or } E(NF_n) = \frac{2}{3}n + o(n).$$

We can do better still, by observing the packing process as it evolves piece by piece :

(1) $NF_n = NF_{n-1} + \delta_n$

where δ_n is 0 if piece n fitted in the same bin with piece n-1 and 1 if X_n was too large and it overflowed.

Define T_n as the level reached by piece n in its bin (whether it is to be the last piece there or not).

Also

$$(2)\quad \begin{cases} P_n(\ell,x) = \mathrm{Prob}(NF_n = \ell,\ T_n \le x) \\ p_n(\ell,x) = d\,P_n(\ell,x)/dx,\ p_n(\ell) = P_n(\ell,1) \\ F_{T_n}(x) = \mathrm{Prob}(T_n \le x) \text{ marginal distribution} \\ f_{T_n}(x) = dF_{T_n}(x)/dx \end{cases}$$

Then $p_1(\ell,x) = f_\chi(x)\delta_{1\ell}$

$$p_n(\ell,x) = \int_{s=0}^{x} p_{n-1}(\ell,s)\, f_\chi(x-s)ds + f_\chi(x) \int_{s=1-x}^{1} p_{n-1}(\ell-1,s)ds \quad n\ge 2$$

and since $f_\chi(u) = 1 \quad 0 \le u \le 1$

$$(3)\qquad p_n(\ell,x) = P_{n-1}(\ell,x) + p_{n-1}(\ell-1) - P_{n-1}(\ell-1,1-x)$$

The marginal equation for T_n is

$$f_{T_1}(x) = f_\chi(x) = 1$$

$$f_{T_n}(x) = f_\chi * f_{T_{n-1}} + f_\chi(x)\,[1 - F_{T_{n-1}}(1-x)]$$

$$(4)\qquad = F_{T_{n-1}}(x) + 1 - F_{T_{n-1}}(1-x)$$

Equation (4) admits the solution

$$(5)\quad f_{T_n}(x) = \begin{cases} 1 & n = 1 \\ 2x & n > 1 \end{cases} \qquad 0 \le x \le 1$$

Now, since

$$E(\delta_n) = \int_{s=0}^{1} f_{T_{n-1}}(s)\, P(\chi_n > 1-s)ds$$

$$= \int_0^1 1.sds = \frac{1}{2} \qquad n = 2$$

$$= \int_0^1 2s.sds = \frac{2}{3} \qquad n > 2$$

we immediately get

$$(6)\quad E(NF_n) = \begin{cases} 1 & n = 1 \\ \frac{2}{3}n + \frac{1}{6} & n > 1 \end{cases}$$

Equation (3) can also be easily solved for the exact distribution of NF_n. For $\chi_i \sim U(0,a)$ with a smaller than the bin size it is much harder to evaluate NF_n, though the mean value can be computed. The interest in this result is that $E(NF_n)$ turns out not to be monotone in a ! There is a local maximum near a = 0.84. See Karmarkar (1982). This curiosity was first noticed through simulation studies. A survey of the state-of-the-art in bin packing analysis, several generalizations and an extensive bibliography may be found in Coffman et al. (1983).

4.2 - Random Access Communication

We turn now to an entirely different computational activity : communications. Specifically we consider a communication network connecting computers which can be characterized as follows :

(a) All the machines have receive-transmit access to a single common error free channel. Propagation time of the signal is negligible (as is the case if all the machine are in a single building or campus). The latter assumption is not essential, but it will simplify the discussion.

(b) There is no other commmunication possible between the users -all coordination must be based on using the common channel.

(c) Communication is done in "packets" - fixed length messages. The users are synchronized, so that the time may be considered to consist of successive slots, the length of slot being equal to the time required to broadcast a packet. Transmission always starts on slot boundary.

(d) All users "listen" continuously to the channel. By slot end each knows whether that slot were idle (no one transmitted), a "success" slot (one user transmitted) or a collision slot (≥ 2 simulataneous broadcasts).

At the present level of discussion the only need for control, or a computational algorithm is to resolve the mentioned collisions, since colliding messages cannot be received correctly and must be retransmitted. In order to analyze any Collision Resolution Algorithm (CRA) we need an assumption about the user population, and we postulate the following :

(e) The process of new message creation may be taken as Poisson, being due to a large number of rarely transmitting users.

One of the more interesting types of CRA is called variously tree algorithm, or stack algorithm. We shall describe a specific CRA that uses a stack.

Since all users are assumed to possess the same system-wide information, besides their own, and to perform the same algorithm, the specification of any CRA must contain a random element - otherwise colliding transmitters will continue to collide ad infinitum. The random element can be the flipping of a coin (simulated on a computer via a pseudo-random number generator), or the message generation times... In the following discussion we use the image of coin-flipping, but this is inessential. We now add the assumptions that prescribe the transmitters' actions :

(f) The common channel is either "available" or "contested".

(g) A user with a packet to send will wait if the channel is contested, and will transmit in the next available slot. Such a user is called "active".

(h) Once a message is transmitted successfully, the user that sent it leaves the active state. If the initial transmission caused a collision, it will start executing the CRA, and the channel enters the constested state. It will become available once the CRA has terminated.

CRA Specification :

(i) Every user performing the CRA manages its position in a system-wide conceptual stack. It is in level 0 at the initial collision slot. It will transmit whenever it is at level 0.

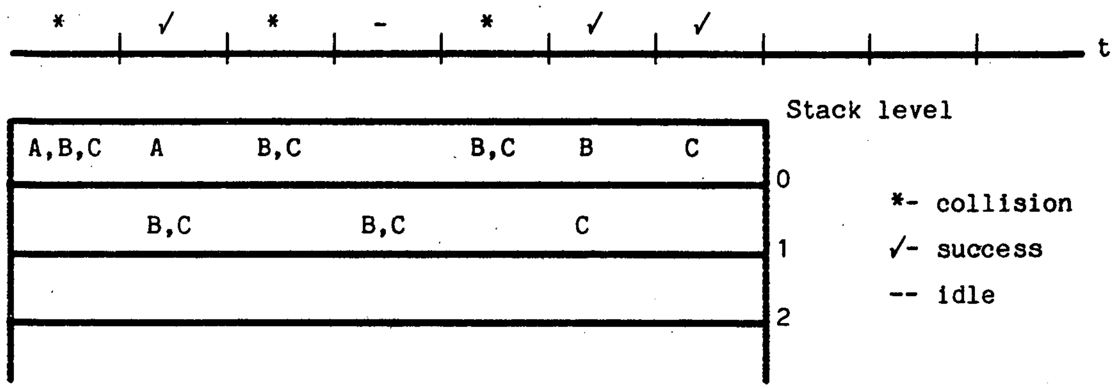
(j) During the Collision Resolution Interval (CRI) the stack level if updated as follows :

(1) At the end of a silent or successful slot decrease the level by 1. In the latter case the successful transmitter quits.

(2) Following a collision, all the users who did not collide (at levels i, i $\geq$ 1) move to level i+1. Each participant in the

collision (i.e. occupants of level 0) remain there with probability p or descend to level 1 with probability, $q = 1-p$.

Example :



In this way the channel alternates between "available" and contested states.

Note that the CRI terminates with probability 1.

The number of users that take part in a collision in the initial slot of the CRI is called the "multiplicity" of the CRI. If we agree to call idle and successful slots that occur during the available phase CRIs of multiplicities 0 and 1 respectively, we can see the entire time axis as consisting of successive CRIs.

Let N_i be the multiplicity of the i-th CRI, and L_{N_i} its duration, then

$$N_{i+1} \sim \mathrm{Pois}(\lambda, L_{N_i})$$

and clearly $\{N_i,\ i \geq 0\}$ form a Markov chain over the non-negative integers. The chain is clearly irreducible and aperiodic. Does it have a non-zero limiting distribution ? Consider the ratio $c_n = \frac{n}{E(L_n)}$.

From standard results concerning Markov chain (e.g. Pakes (1969) and Kaplan (1979)) it follows that the chain will be stable for $\lambda < \liminf c_n$ and unstable for $\lambda > \limsup c_n$. It only remains to evaluate these ratios...

The stack formulation is easily transcribed into a recursive relation :

$$(6)\quad L_n = \begin{cases} 1 & n = 0,1 \\ 1+L_I+L_{n-I} & n > 1 \end{cases}$$

where the three components in the relation for $n > 1$ are the initial collision slot, the time required to clear the packets remaining at level 0 (if any), and the time required to transmit successfully those that elected level 1 (if any), respectively. The variable I has the distribution $B(n,p)$.

Defining the probability functions (pgf's)

$$g_n(u) = \sum_{i\geq 1} P(L_n=i)u^i$$

$$h(u,z) = \sum_{n\geq 0} g_n(u)z^n/n!$$

we successively obtain

$$(7)\quad g_n(u) = \begin{cases} u & n = 0,1 \\ u\sum_{i=0}^{n} \binom{n}{i} p^i q^{n-i}\, g_i(u)\, g_{n-i}(u) & n > 1 \end{cases}$$

and

$$(8)\quad h(u,z) = u(1-u^2)(1+z)+uh(u,pz)h(u,qz).$$

The analytic properties of $h(.,.)$ are not easy to glean from (8). It is remarkably similar to equations arising in the investigations of multipype branching processes; what we designate as "collision multiplicity" would serve there as "descendant type". The objectives of the computations are however rather different, and so are the required treatments.

Note that equation (8) is really an equation in one variable, as u can be considered there a mere parameter.

To determine the channel capacity only the expected values $E(L_n)$ are required, however. Differentiating (8) with respect to u, at $u = 1$, and denoting

$$\alpha_n = E(L_n),\ \alpha(z) = \sum_{n\geq 0} \alpha_n z^n/n!$$

we obtain

$$(9)\quad \alpha(z) = e^z-2(1+z)+e^{qz}\alpha(pz)+e^{pz}\alpha(qz)$$

and introducing $\phi(z) = e^{-z}\alpha(z)$.

(10) $\phi(z) - \phi(pz) - \phi(qz) = 1-2(1+z)e^{-z}$.

Equation (10) is a convenient starting point, though not necessarily the only one. Note that directly from (6) one can write the following relation between the expectations, which is equivalent to (10) :

(11) $\alpha_n = 1 + \sum_{i=0}^{n} \binom{n}{i} p^i q^{n-i} (\alpha_i + \alpha_{n-i})$.

And furthermore, equation (11) can be readily reduced to a recursion :

(12) $\alpha_n = [1+p^n+q^n+\sum_{i=1}^{n-1} \binom{n}{i} p^i q^{n-i} (\alpha_i + \alpha_{n-i})] \;/\; (1-p^n-q^n)$.

Equation (10) can be used to obtain a closed form expression for α_n : Let $\phi(z) = \sum_{n\geq 0} \phi_n z^n$; equating coefficients of powers of z in (10) we get

(13) $$\phi_n = \begin{cases} 1 & n = 0 \\ \dfrac{2(-1)^n(n-1)}{n!(1-p^n-q^n)} & n > 1 \end{cases}$$

The same relation follows from taking the Mellin transform of (10). The relation $\alpha(z) = e^z\phi(z)$, in terms of the generated sequences is

$$\alpha_n = n! \sum_{k=0}^{n} \frac{\phi_k}{(n-k)!}$$

and thus

(14) $$\alpha_n = 1 + \sum_{k=2}^{n} \binom{n}{k} \frac{2(-1)^k(k-1)}{1-p^k-q^k}. \quad n \geq 0$$

Differentiating equation (14) twice with respect to p, it is easy to see that the only solution of the equation $\alpha'_n = 0$ for $p \in (0,1)$ is $p = 1/2$ and that the second derivative indeed is positive there, thus this value minimizes α_n, as intuition and symmetry might had led one to believe. The optimum is very flat though, and typically α_n varies by less than one percent where p varies in (0.44, 0.56).

The explicit relations in eq.(14) while of interest in their own right are not satisfactory for the analytical determination of the channel capacity. We need an explicit expression of the dependence of α_n on n, though an asymptotic evaluation would suffice. It is not surprising that surprising that methods that have been found useful in dealing with the expected complexity of graph algorithms are found just as good in treating an algorithm that was originally presented as the "serial tree algorithm".

We begin with equation (14) which can be rewritten as

$$(15)\qquad \begin{aligned}\alpha_n &= 1 + 2\sum_{k=2}^{n}\binom{n}{k}(k-1)(-1)^k \sum_{i\geq 0}(p^k+q^k)^i \\ &= 1 + 2\sum_{i\geq 0}\sum_{r=0}^{i}\binom{i}{r}\sum_{k=2}^{n}\binom{n}{k}(k-1)\beta^k, \quad \beta = -p^r q^{i-r}\end{aligned}$$

Summing over k and shifting the n subcript one obtains

$$(16)\qquad \alpha_{n+1} = 1 + 2\sum_{i\geq 0}\sum_{r=0}^{i}\binom{i}{r}\left[(1+\beta)^n n\beta-(1+\beta)^n+1\right]$$

Writting $x = -n\beta$ we approximate, for large n, $(1+\beta)^n = (1 - \frac{x}{n})^n$ by e^{-x}. It can be shown that the approximate value, a_n differs from the original α_n by only $O(1)$. (Though for the purpose at hand an approximation to within $O(n^{1-\epsilon})$, for some $\epsilon > 0$ would be adequate). See Hofri (1984). We proceed then with

$$(17)\qquad \alpha_{n+1} = 1 + 2\sum_{i\geq 0}\sum_{r=0}^{i}\binom{i}{r}\left[1 - xe^{-x}-e^{-x}\right] \quad x = np^r q^{i-r}$$

A standard relation for the gamma function is*

$$e^{-x} = \int_{(c)} \Gamma(z)x^{-z}dz \qquad c > 0$$

and moving the contour of integration past the poles of $\Gamma(z)$ at $z = 0,-1$ one gets

$$e^{-x} + x - 1 = \int_{(c)} \Gamma(z)x^{-z}dz \qquad -2 < c < -1$$

which is precisely what we need in order to rewritte (17), taking, arbitrarily, the midpoints in the integration strips :

$$(18)\qquad \begin{aligned}a_{n+1} &= 1-2\sum_{i\geq 0}\sum_{r}\binom{i}{r}\left[x(e^{-x}-1)+(e^{-x}+x-1)\right] \\ &= 1-2\sum_{i\geq 0}\sum_{r}\binom{i}{r}\left[x\int_{(-1/2)}\Gamma(z)x^{-z}dz + \int_{(-\frac{3}{2})}\Gamma(z)x^{-z}dz\right]\end{aligned}$$

* The notation $\int_{(c)}$ stands for $\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}$

The second integral of (18) is, by change of variable, equal to

$$\int_{(-1/2)} \Gamma(z-1)x^{1-z}dz = \int_{(-1/2)} \frac{\Gamma(z)}{z-1}x^{1-z}dz, \text{ hence}$$

$$a_{n+1} = 1-2\sum_{i\geq 0}\int_{(-1/2)} \Gamma(z)\frac{z}{z-1}\sum_{r=0}^{i}\binom{i}{r} x^{1-z}dz$$

$$= 1-2\sum_{i\geq 0}\int_{(-1/2)} \Gamma(z)\frac{zn^{1-z}}{z-1}q^{i(1-z)}(1+(\frac{p}{q})^{1-z})^{i}dz$$

$$(19) \qquad = 1-2\int_{(-1/2)} \Gamma(z)\frac{z}{z-1}\frac{n^{1-z}}{1-p^{1-z}-q^{1-z}}dz$$

Note that the last summation converges only if $Re(z) < 0$, hence the special representation chosen in equation (18). The merit of equation (19) over all preceding steps is that n is exposed here "in splendid isolation". The evaluation of the contour integral is routine : one goes from $(-1/2,-iN_1)$ to $(-1/2, iN_1)$ to (N_2,iN_1) to $(N_2,-iN_1)$ to $(-1/2,-iN_1)$ -a negative sense. The contribution of the horizontal parts of the contour decays when N_1 increases becauce $|\Gamma(t+iN)| = O(|t+iN|^{t-1/2}e^{-t-\pi N/2})$, and the n^{1-z} takes care of the receding vertical component. Hence the required integral is minus the sum of residues of the integrand to the right of the contour. Determining these requires information about the roots of $1-p^{s}-q^{s}$. For a few isolated values of p these can be determined*, but in general all one can say is : a) $Re(s) \leq 1$ and for the most part $-1 < Re(s)$, with rather rare exceptions. b) The roots are well separated and thus easy to determine numerically.

The pole at $z = 0$ is the first to consider; it is simple and the residue there contribute $\frac{-n}{p\log p + q\log q}$; the rest we list implicity, yielding :

$$(20) \qquad \alpha_n = 1+2n\ [\frac{-1}{p\log p + q\log q} + 2Re\{\sum_{\zeta}\frac{\zeta\Gamma(\zeta)}{\zeta - 1}n^{-\zeta}Res[1-p^{1-z}-q^{1-z}]^{-1}_{z=\zeta} + O(1)$$

where 'Res' denotes the residue of the quantity in brackets at the designated point, and ζ goes over all roots of $1-p^{1-z}-q^{1-z}$ that lie to the right of the contour. The sum in the braces turns out to be extremely small, due to cancellations which are quite hard to estimate analytically. It is typically four to seven orders of mangnatude below the leading term.

---

*E.g. for $p=1/2$ the roots are $s=1+2\pi ik/\log 2$ k being any integer; for $p=2/(1+\sqrt{5})=\phi^{-1}$, the golden ratio, $s=(-1)^{k+1}+k\pi i/\log\phi$.

A subsidiary result from the above, that may be unsettling at first is that while α_n is essentially linear in n, the quantity α_n/n does not approach a limit as $n \to \infty$, but oscillates with a minute amplitude and ever increasing period around the value of the leading term above, that is fixed in n.

The main information supplied then by equation (20) is that

$$(21) \quad 1/c_n(p) \equiv E(L_n)/n = \frac{-2}{p\log p + q\log q} + d_n(p)$$

where $d_n(p)$ is extremely small for all n and p, with a maximal value over n of $\bar{d}(p)$.

Example : for $p=1/2$ the leading term is $c^{-1} = \frac{2}{\log 2} \approx 2.8854$, and $\bar{d}(1/2) \approx 6.10^{-5}$.

Thus we obtain that for $p = q = 1/2$

$$(22) \quad \lambda_{max} = \lim\inf c_n^{-1} \approx c^{-1} - \bar{d}(1/2) = 0.346559.$$

The above algorithm is the simplest one in a large (and growing) family of CRAs, that by dint of clever use of the feedback can be stable with higher traffic rate. The highest rate known at this time is 0.4877. By mixing these algorithms with reservation schemes rates exceeding 0.7 have been shown to be achievable. The best access to the state-of-the-art to this area at this time is probably through the "Special Issue on Random Access Communications", Guest Editor J.L. Massey, IEEE Trans. Inf. Theory, IT-31 #2 (1985).

REFERENCES

COFFMAN E.G. Jr., GAREY M.R., JOHNSON D.S. : Approximation Algorithms for Bin-Packing - An Updated Survey (1983). Bell Labs, Murray Hill, NJ 07974. Copies are available from the authors.

FAYOLLE G., FLAJOLET P., HOFRI M., JACQUET P. : Analysis of a Stack Algorithm for Random Multiple-Access Communication. IEEE Trans. on Inf. Th., IT-31 #2 (1985).

FISHMAN G.S. (1978) : Principles of Discrete Event Simulation, J. Wiley, New York.

HAMMERSLEY J.M., HANDSCOMB D.C. (1964) : Monte-Carlo Methods Methuen, London.

HOFRI M. : Stack algorithms for collision detecting channels and their analysis in "Modelling and Performance Evaluation Methodology, Proc. of the Int. Sem. Paris, France, January 1983". pp.71-88. Springer-Verlag Lecture Notes in Control and Inf. Sciences, 1984.

KAPLAN M. : A Sufficient condition for nonergodicity of a Markov chain. IEEE Trans. on Inf. The. IT-25 # 4, pp.470-471 (1979).

KNUTH D.E. (1972) : The Art of Computer Programming, vol 2 : Semi-numerical Alogorithms. Addison-Wesley, Reading MA.

PAKES A.G. : Some conditions for ergodicity and recurrence of Markov chains. Opns. Res. 17 pp. 1058-1061 (1969).

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique